yanis Mzyene
yanis Maatouk

Rapport de Projet : Analyseur Syntaxique

---

1. Introduction

Le but de ce projet est de créer un analyseur syntaxique pour un petit langage de programmation, appelé **TPC**, en utilisant les outils **Flex** et **Bison**. Le langage **TPC** est un sous-ensemble simplifié du langage **C**, et l'objectif est de détecter et traiter les erreurs lexicales et syntaxiques dans un programme source en TPC, tout en générant un arbre abstrait pour les programmes valides.

---

2. Définition du Langage Source

Le langage **TPC** est conçu autour de la gestion de fonctions et de variables, et utilise les types de base suivants :

- **int** (entier signé)
- **char** (caractère)

Le mot-clé **void** est utilisé pour indiquer une fonction sans retour ou sans argument.

Les programmes **TPC** sont composés de :

- **Déclarations de variables**
- **Fonctions** (avec leurs déclarations de variables locales et leurs instructions)

Des variables globales peuvent également être déclarées avant les fonctions.

---

3. Lexèmes du Langage

Le langage **TPC** utilise les lexèmes suivants :

- **Identificateurs** : Composés de lettres, chiffres et symboles de soulignement (_), avec distinction entre majuscules et minuscules.
- **Constantes numériques** : Suites de chiffres.
- **Caractères littéraux** : Délimités par des apostrophes (') comme en C.
- **Commentaires** : Soit délimités par `/*` et `*/`, soit par `//` et la fin de ligne.
- **Opérateurs** : `=`, `+`, `-`, `*`, `/`, `%`, `!`, `==`, `!=`, `<`, `>`, `<=`, `>=`, `&&`, `||`.
- **Délimiteurs** : `;`, `,`, `()`, `{}`, etc.

Les mots-clés comme **if**, **else**, **return**, etc., sont réservés et ne peuvent pas être utilisés comme identificateurs.

---

4. Grammaire du Langage TPC

La grammaire de **TPC** est définie en **Bison** pour générer un analyseur syntaxique. Elle permet de structurer les programmes en déclarations de variables, déclarations de fonctions, et instructions. Les règles syntaxiques incluent :

- **Déclaration de variables** : Déclarations locales et globales.
- **Déclaration de fonctions** : Avec ou sans retour.
- **Instructions** : Affectations, appels de fonctions, expressions conditionnelles, etc.

---

5. Travail Demandé

L'objectif principal de ce projet est de :

- Étendre la grammaire et les lexèmes pour gérer les variables locales **statiques** (comme `static int language;`).
- Implémenter un analyseur syntaxique avec **Flex** et **Bison**.
- Générer un arbre abstrait pour les programmes valides et afficher cet arbre.

---

6. Organisation du Projet

Le projet est organisé dans les répertoires suivants :

- **src** : Contient les fichiers source, y compris les fichiers Flex et Bison.
- **bin** : Contient le binaire de l'analyseur `tpcas`.
- **obj** : Fichiers intermédiaires.
- **test** : Contient les jeux d'essais, avec deux sous-répertoires :
    - **good** : Programmes valides.
    - **syn-err** : Programmes contenant des erreurs.
- **rep** : Contient ce rapport.
- **racine :** Contient le makefile ainsi que le script bash.

Le programme **tpcas** peut être exécuté via la ligne de commande avec les options :

- `-t` ou `--tree` : Affiche l'arbre abstrait.
- `-h` ou `--help` : Affiche l'aide et termine l'exécution.

---

7. Détails de l’Implémentation

7.1. Analyse Lexicale (Flex)

L’analyse lexicale utilise **Flex** pour identifier les différents lexèmes dans le programme **TPC**. Les règles lexicales gèrent les identificateurs, les constantes numériques, les caractères, et les opérateurs. Les commentaires sont correctement ignorés.

7.2. Analyse Syntaxique (Bison)

L’analyse syntaxique est implémentée avec **Bison**, en suivant la grammaire définie. Les règles de la grammaire permettent de construire l’arbre abstrait pour les programmes valides.

---

8. Problèmes Rencontrés et Solutions

8.1. Gestion des Commentaires

Les commentaires multi-lignes ont posé un défi en raison de la gestion du contexte d’ouverture et de fermeture. Nous avons résolu ce problème en utilisant les états dans **Flex** (`COM1`, `COM2`).

8.2. Extensions du Langage

L’extension du langage pour inclure des variables statiques a nécessité des ajustements dans la grammaire et le code, notamment pour ne pas permettre l’utilisation de `static` dans les déclarations globales.

8.3. Gestion des Copies des Variables

Un problème a été rencontré concernant la gestion des copies des variables, qui ne s'affichaient pas correctement sur l'arbre abstrait généré. Ce problème était dû à une mauvaise gestion des valeurs assignées aux variables dans certaines parties du code. Après des ajustements, nous avons corrigé la manière dont les variables étaient traitées, ce qui a permis une représentation correcte des copies dans l'arbre abstrait.

---

9. Conclusion

Ce projet permet de créer un analyseur syntaxique fonctionnel pour le langage TPC en utilisant Flex et Bison. L'analyseur est capable de détecter les erreurs lexicales et syntaxiques et de générer un arbre abstrait pour les programmes valides. Des améliorations peuvent être envisagées pour gérer d'autres extensions du langage, et cet analyseur pourrait servir de base pour un compilateur.